Some Results from a New Technique for Response Time Estimation in Parallel DBMS

Author Details

Neven Tomov, Euan Dempster (euan@cee.hw.ac.uk, corresponding author), M. Howard Williams (presenting author), Albert Burger, Hamish Taylor, Peter J. B. King

Department of Computing and El. Engineering
Heriot-Watt University
Edinburgh
EH14 4AS
UK

Phil Broughton

International Computers Limited
High Performance Technology
Wenlock Way
West Gorton
Manchester
M12 5DR
UK

Some Results from a New Technique for Response Time Estimation in Parallel DBMS

Neven Tomov[1], Euan Dempster[1], M. Howard Williams[1], Albert Burger[1], Hamish Taylor[1], Peter J.B. King[1], and Phil Broughton[2]

[1] Department of Computing and El. Engineering,
Heriot-Watt University, Edinburgh EH14 4AS, UK
[neven,euan,howard,ab,hamish,pjbk]@cee.hw.ac.uk
[2] International Computers Limited,
High Performance Technology,
Wenlock Way, West Gorton, Manchester M12 5DR, UK
pb@wg.icl.co.uk

Abstract. The need for tools for performance prediction of parallel database systems is generally recognised. One such tool which has been developed (Steady) is based on analytical techniques to obtain a rapid estimate of performance. The approach to predicting response time involves a heuristic approximation coupled with standard queueing solutions. This paper reports on preliminary results for both maximum transaction throughput and response time obtained in comparing this approach against actual measurements.

1 Introduction

The growth of commercial interest in the potential and use of parallel computers for running relational database applications has been noted by Norman and Thanisch [1]. Such applications offer rich amounts of exploitable parallelism that database management systems running on parallel platforms can take advantage of. Several well known commercial DBMSs such as Oracle [2], Informix [3], Ingres [4], Sybase [5] and DB2 [6] are now available on popular and dedicated SMP and MPP machines.

The ability to predict the performance of parallel relational databases is important for their application sizing, capacity planning, data placement and performance tuning. To assist in these processes, an analytical technique for estimating the performance of parallel relational database systems has been developed. The technique has been incorporated in a tool [7, 8] which can rapidly estimate how relational database applications will behave without running lengthy simulations or actually trying them out.

Part of the performance estimation technique involves prediction of query response time, which is a non-trivial problem. This paper reports on experiments in which the results of the technique are compared with actual measurements obtained from a parallel platform. The experiments are performed as part of the process of validation of the analytical approach.

The remainder of this paper is organised as follows. Section 2.1 briefly describes the hardware platform (the ICL GoldRush MegaSERVER [9]) and DBMS (Informix XPS [3]) experimental setup. This is followed by a description in section 2.2 of a subset of the queries and data used in the comparisons. Section 2.3 provides a brief introduction to Steady – the parallel DBMS performance prediction tool. The response time prediction technique incorporated in Steady is discussed in section 2.4. Section 2.5 explains the methods used in obtaining performance figures. Some comparison graphs of actual vs. predicted transaction throughput and query response time are given in section 3. Finally, section 4 provides a summary and conclusions.

2 Obtaining Performance Measures

The ICL GoldRush MegaSERVER [9] is a parallel platform developed as a backend database server to host several different database systems (Ingres, Oracle, Informix). Steady is a performance prediction tool designed specifically for shared-nothing parallel architectures [10] and calibrated for the ICL GoldRush platform running Informix XPS.

2.1 GoldRush and Informix XPS

The basic GoldRush hardware architecture consists of a number of Processing Elements (PEs) and Communication Elements (CEs) linked by a high speed DeltaNet network. Each PE is connected to its own disc storage subsystem and runs a UNIX operating system and DBMS code. A CE provides external links for GoldRush to clients via LANs. The particular GoldRush setup used here has 1 CE, 8 PEs, 6 discs per PE and a cache of 16MBytes on each PE.

This type of architecture is an ideal platform for the Informix Extended Parallel Server(XPS) [3]. It contains a set of internal components called co-servers which are installed on each of the PEs of GoldRush. A co-server provides a user entry point to GoldRush on a given PE. Locks on data items and all data processing are managed locally within each co-server. Where deemed suitable, single queries are broken into subtasks and processed concurrently by threads within a single co-server and across co-servers. Data can be partitioned across discs and PEs so that parallel I/O operations can take place. Additionally, certain partitions, known to be irrelevant for a particular query, can be skipped altogether.

2.2 Tables and Queries

The experiments detailed in this paper are carried out on a subset of the AS3AP benchmark [11] tables. In particular, a number of variations of the *uniques* relation are used; the attributes of *uniques* are given in Table 1.

There are four *uniques* relations used. The first is called *uniques30k*. It has 30000 tuples and is placed on one disc of one PE (PE 7). The second relation

Table 1. Attributes of AS3AP *uniques* relation

key	integer(4)	decim	numeric(18,2)
int	integer(4)	date	datetime(8)
signed	integer(4)	code	char(10)
float	real(4)	name	char(20)
double	double(8)	address	varchar(20)

is called *uniques80* and has 80 tuples which are also stored on one disc of PE 7. The other two relations are *uniques270k* and *uniques540k*. They contain 270000 and 540000 tuples respectively and are placed across all eight PEs. Each of these two tables is fragmented into 8 fragments by a simple hash function on the *key* primary key attribute with one hash fragment placed on a single disc of each of the PEs. Each tuple has a unique value for attributes *key* and *int*. Attribute *signed*, however, is modified from the benchmark specification so that tuples have *signed* values in the range 1 to 10 with an equal number of tuples for each value.

A number of experiments have been performed and two will be presented here as typical. The first query performs a simple aggregation of a *uniques* relation, looking for the maximum and minimum values of the *int* attribute, for those tuples whose *signed* value is not 1. Thus the aggregation is carried over $9/10^{ths}$ of the *uniques* tuples. Tables *uniques30k* and *uniques270k* are used:

SELECT max(*int*), min(*int*)
FROM *uniques30k* (*uniques270k*)
WHERE *signed* not in (1)

The second query finds the maximum value of the *int* attribute from the result of a join of *uniques80* with *uniques540k* where the *int* values are the same. The tuples of *uniques80* are taken from *uniques540k* so that the size of the resulting join is 80 tuples:

SELECT max(*uniques80.int*)
FROM *uniques80*, *uniques540k*
WHERE *uniques80.int* = *uniques540k.int*

Although this could be handled with a much simpler query, this form was used in order to perform a join of this type and which would facilitate measurements of this operation.

Table *uniques30k* is placed on one PE and queries involving only it are not parallelised. Those involving tables *uniques270k* and *uniques540k*, however, are performed in parallel by all co-servers containing the table data. The second query employs a hash-join algorithm, which builds a hash table across all co-servers using *uniques80*, and subsequently probes this with the tuples from *uniques540k*.

2.3 Steady

STEADY [7, 8] (System Throughput Estimator for Advanced Database sYstems) is an analytical tool for performance estimation of parallel relational database systems. It has been designed to handle a range of different platforms although thus far it has only been calibrated against the GoldRush platform. Apart from a graphical user interface, it consists of five major modules:

1. The **Profiler** is a statistical tool primarily responsible for generating base relation profiles and estimating the number of tuples resulting from data operations;
2. **DPTool** is used to generate various data placement schemes for a parallel database using different strategies;
3. The **Modeller** is responsible for producing the profile of the tasks required for a particular benchmark or query with assistance from the Profiler, query paralleliser and the cache model [12];
4. The **Evaluator** takes the task profiles and produces resource usage profiles, maximum system throughput values and system bottlenecks;
5. The **Response Time Estimator** takes the resource usage profiles and from these estimates response times.

Relations are partitioned into fragments by DPTool using declustering methods such as *hash*. These fragments are then allocated to the PEs using placement methods such as *size, bubba, hua, etc.*. These relation fragments can then be assigned to the discs of PEs according to various methods. Based on the generated data placement and the chosen DBMS architecture, a profile of the tasks required for a particular benchmark or query is generated by the Modeller which includes an estimation of disk I/O requirements in terms of the number of pages read and written to each disk. This is derived on the basis of the estimated cache hit ratios for the particular relation fragments in the cache of each PE and is carried out by the cache model. The task profiles are then converted by the Evaluator into resource utilisation profiles, from which the system bottlenecks and the maximum throughput rate are determined. This gives the user an indication of the upper limit of system capacity in terms of throughput. The resource usage blocks are then fed into the Response Time Estimator which estimates the response time of the query, as described briefly in the following section.

2.4 Response Time Technique

The input to the Response Time Estimator module of Steady consists of the original query task profiles represented as patterns of resource consumption incurred on the various GoldRush resources such as CPUs, discs, and interconnect. This representation is equivalent to an open multi-class queueing network [13]. In general, however, the networks obtained are not in product form, due to the non-exponential service times required at each resource. This means that exact solutions for quantities such as the mean response time cannot be obtained analytically.

To overcome this problem, a heuristic is proposed in [14]. It specifies a procedure for labeling each resource in the queueing network as either an M/M/1 or an M/G/1 resource. The decision is based on knowledge of the resource's utilisation and relative visit ratio [13]. With each resource labeled, the network can be solved to obtain the mean response time of individual queries.

2.5 Taking Measurements

Calibration of the models required running a selection of queries, using the parallelised query execution plans to determine the way in which they were broken down by the system, and reconciling these against measurements obtained using the Informix XPS performance measuring tool *onstat*.

Once this was complete, a transaction generator was created to emulate a parallel database system workload with many users independently querying the data. It was used to fire single-query transactions from the communication element (CE) to a given co-server at a specified rate. Two generators were created. The first fired transactions with constant inter-arrival times and was used to determine the arrival rate for which the maximum throughput of the machine can be achieved.

The second version of the transaction generator fired transactions with exponentially distributed inter-arrival times and was used to obtain transaction response times.

Figure 1 shows the response times for query 1 on *uniques30k*. The query is fired 100 times with exponentially distributed inter-arrival times at a rate of 0.04 transactions per second which corresponds to 11% of the predicted maximum throughput.

3 Results

Figures 2 and 3 show the throughput results for query 1 with the 30K tuple table and the 270K tuple table, respectively. Note that since the 270K table is spread across 8 nodes (i.e. each node has approximately 30375 tuples) the performance achieved is similar to that for the 30K table which is placed on a single node. Figure 4 shows the results for query 2. Similarly, Figures 5 and 6 show the response time results for query 1 with the 30K tuple table and the 270K tuple table, respectively. Figure 7 shows the response time results for query 2.

The figures show that the maximum throughput achieved on GoldRush is slightly less than that predicted by Steady by about 13%. We believe that this is due to additional operating system overhead which we were not able to isolate during the calibration process. Similarly response times observed were slightly larger than those predicted by Steady — less than 20% for arrival rates less than 70% of maximum throughput (as predicted by Steady), and less than 30% for arrival rates less than 90% of maximum throughput.

A number of other examples which have been investigated produce results which are similar to those for the queries presented here. Error margins are roughly the same for the cases investigated so far.

Fig. 1. Individual response times for 100 queries of type 1 with a table of 30K tuples. Query arrivals are Poisson with rate 0.04 (11% of predicted max throughput).

4 Conclusions

This paper has reported on experiments in which the results of an analytical technique for performance estimation in parallel relational DBMS are compared with actual measurements obtained from a parallel platform. Both throughput and response time results are presented.

The results are encouraging and generally the relative difference between the measured and predicted performance is less than 20%. The relative error in the predictions is largest for arrival rates corresponding to a bottleneck resource utilisation of 80% and above. In practice, however, one is not likely to load the system much beyond 70% utilisation – thus 80% is a reasonable upper limit for performance prediction beyond which queues rapidly become unmanageable. The fact that Steady tends to predict lower response times and higher maximum throughputs can be attributed to additional resource consumption by the OS which is presently not taken into account by the analytical model.

Fig. 2. Throughput of query 1 with a table of 30K tuples.

Acknowledgements

The authors acknowledge the support received from the UK Engineering and Physical Sciences Research Council (EPSRC) under the PSTPA programme (GR/K40345) and from the Commission of the European Union under the Framework IV programme (Mercury project). They also wish to thank Arthur Fitzjohn and Monique Mitchell of ICL(Manchester).

References

1. Norman, M.G., Thanisch., P., Parallel Database Technology, Bloor Research Group, Netherlands (1995) 1-546
2. Oracle Corporation., Oracle7 Parallel Server Concepts & Administration, Release 7.3, Oracle, Server Products, California, USA (1996)
3. Informix Software Inc., Informix OnLine Dynamic Server administrator's guide (1994)
4. The ASK Group Inc., ASK OpenINGRES database administrator's guide (1994)
5. Sybase Inc., Sybase, http://www.sybase.com (1998)
6. IBM Corp., DB2, http://www.software.ibm.com/data/db2 (1998)

Fig. 3. Throughput of query 1 with a table of 270K tuples.

7. Zhou, S., Williams, M.H., Taylor, H., Practical throughput estimation for parallel databases, IEE Software Engineering Journal (July 1996)
8. Williams, M.H., Dempster, E.W., Tomov, N.T., Pua, C.S., Taylor, H., Burger, A., Lü, J., Broughton, P., An Analytical Tool for Predicting the Performance of Parallel Relational Databases, submitted to Concurrency: Practice and Experience (1998)
9. Watson, P., Catlow, G., The architecture of the ICL GoldRush MegaSERVER, Proceedings of the 13th British National Conference on Databases (BNCOD 13) Manchester U.K. (July 1995) 250–262
10. Stonebraker, M., The case for shared nothing, Database Engineering 9(1) (March 1986)
11. Turbyfill, C., Orji, C., Bitton, D., AS3AP: An ANSI SQL Standard Scaleable and Portable Benchmark for Relational Database Systems, The Benchmark Handbook, Second Edition (1993), Gray, J. (editor)
12. Zhou, S., Tomov, N., Williams, M.H., Burger, A., Taylor, H.: Cache Modelling in a Performance Evaluator of Parallel Database Systems Proceedings of the Fifth International Symposium on Modelling, Analysis and Simulation of Computer and Telecommunication Systems (January 1997) 46-50
13. Molloy, M.: Fundamentals of Performance Modeling Macmillan Publishing Company (1989)

Fig. 4. Throughput of query 2 with a table of 540K tuples.

14. Tomov, N.T., Dempster, E.W., Williams, M.H., King, P.J.B., Burger, A.: Approximate Estimation of Transaction Response Time, to appear in The Computer Journal

Fig. 5. Response time of query 1 with a table of 30K tuples.

Fig. 6. Response time of query 1 with a table of 270K tuples.

Fig. 7. Response time of query 2 with a table of 540K tuples.